QoS-Based Communication Protocols in Wireless Sensor Networks

Serdar Vural, Yuan Tian, and Eylem Ekici
Department of Electrical and Computer Engineering
Ohio State University, Columbus, OH 43210
Email: {vurals,tiany,ekici}@ece.osu.edu

I. Introduction

Wireless Sensor Networks (WSN) have been one of the main research foci in wireless networks area over the last decade [2]. With the evolution of the MEMS technology and the availability of low cost communication and computation hardware, WSNs have been transformed from conceptual paradigms to reality in this short period. Prototype and deployed WSNs currently serve as enablers of several applications such as environmental monitoring, urban safety, traffic monitoring, smart spaces, and surveillance of hostile and inaccessible areas. The majority of the early research efforts have focused on enabling technologies for WSNs, including development of communication protocols, localization methods, and simple application-dependent information processing.

Major research efforts in WSN area have resulted in many deployed testbeds and other implementations. The driving motivation of delivering solutions that can be implemented in short time resulted in systems that can provide only best effort service. Minimization of energy consumption or achieving "high efficiency" (in its versatile definition) has been the objective of many communication protocols designed for WSNs. Their performance strictly depends on the configuration of the network and the load it carries. While the boosting effect of the existing solutions on the research community should be acknowledged, these solutions fall short of addressing requirements of *all* WSN applications and deployment scenarios. More specifically, mission critical and real-time applications suffer from unpredictable performance levels when such communication protocols are used.

Mission critical and real-time applications require performance guarantees from the system on which they are implemented. As an example, a real-time intrusion detection application running on a WSN may require the event detection decision to be made within a given delay bound [40]. Other applications may require that the network delivers data packets to the sink with a given probability. Similarly, the overall energy consumption of all communication events may be subject to energy consumption constraint. Applications running on Video Sensor Networks [19], [11], [18], which form a new and emerging class of WSNs, inherently require service guarantees from the communication network due to real-time nature of its multimedia source content. These requirements are classified as *Quality of Service* (QoS) in other wired and wireless networks, which we also adopt for WSN environments. We classify a communication protocol as QoS-based protocol if it can guarantee one or more performance metrics to upper layers or the application. Under this classification, solutions that simply minimize/maximize a performance metric (delay, energy consumption, packet loss probability, network lifetime, etc.) without performance guarantees are considered as non-QoS-based solutions. We should note that most of the existing proposals for WSNs fall into non-QoS-based category.

In this chapter, we first contrast QoS provisioning in WSNs and other network types and introduce a QoS provisioning framework for WSNs. Then we outline and discuss proposed QoS-based communication protocols for WSNs. We also outline methods that support QoS-based in-network processing along with communication for WSNs. QoS-based capacity estimation methods are also discussed within the proposed framework. We then conclude the chapter with future research directions.

II. QoS in Wireless Sensor Networks

A. General Principles

QoS has been the target of many communication protocols for numerous network types. In its broadest form, Quality of Service refers to the contract between the service provider, i.e., the network, and customers, i.e., applications [12]. In wired networks, one of the main motivations for QoS solutions is the real-time multimedia applications that need bandwidth, delay, and jitter guarantees. ATM networks [14] were proposed to support such requirements from ground up. Although ATM networks are not as widely in use as originally imagined, QoS support mechanisms proposed for ATM networks still inspire new solutions. In IP networks, QoS support of individual flows have been proposed to be handled through IntServ [10] mechanism, which has not gained wide-spread acceptance due to its scalability problems[1]. In cellular networks, the motivation for QoS support is also inherent to the primary application of such networks: Voice (and recently) video calls are subject to stringent constraints to be commercially viable. In these networks, QoS support are provided through *resource reservation* mechanisms. To accomplish resource reservation, the following steps are followed:

- *Available Resource Estimation:* The first step in QoS support is the knowledge of available resources. The estimation of available resources is performed using the *network state* and the communication protocols employed in the network. The network state is comprised of the network connectivity information, maximum capacity of nodes and links, and allocated resources.
- *Calculation of Required Resources:* Given that the performance requirements of applications are known, resources required to sustain the QoS expectations are calculated in the network. Both performance metric conversion as well as the resource requirement estimation depend on the protocols used in the network. The calculation also involves the selection of the resources in the network for the information flow.
- *Resource Allocation:* Calculated resources are reserved in the network entities. The reservation of such resources is performed via auxiliary protocols such as RSVP [38] or as an integral part of the communication protocol.
- *Resource Deallocation:* When a session terminates, resources are returned to the general pool. The deallocation can be done either explicitly, or implicitly through timeout mechanism.

[1]The DiffServ [7] architecture has been proposed later on to support differentiation of groups of flows rather than individual flows to overcome the scalability problem. However, DiffServ mechanism does not provide absolute performance guarantees and therefore cannot be classified as a QoS solution.

B. QoS in Ad Hoc Networks

The above outlined steps work well in networks where resources are separated from each other with well-defined boundaries: In wired networks, link as well as node resources are clearly separated from each other. As an example, two node-disjoint links can be treated as independent resources even though the load on them may depend on each other. Similarly, in cellular networks, point-to-point links between mobile stations and base stations are separated from each other in time, frequency, code, space, or a combination thereof. Hence, QoS provisioning in both types of networks can easily follow the aforementioned steps.

In wireless ad hoc networks, the resource allocation strategy faces a roadblock at a very fundamental level [16]: Estimation of available resources is a non-trivial task even for simplest multi-hop ad hoc networks. Wireless resources are shared by multiple nodes which do not have an inherent coordination infrastructure. The contention resolution and resource bidding procedures usually propagate over long distances and affect far-away nodes. The lack of isolation of resources and the dynamic nature of the network make resource estimation and allocation very challenging tasks. This fact, coupled with the weak motivation for QoS-demanding applications for ad hoc networks, have limited the acceptance of QoS-based communication proposals for ad hoc networks.

C. QoS in Wireless Sensor Networks

A WSN can be regarded as a special type of ad hoc network with very resource-constrained nodes, lower mobility, and larger scale. With these additional constraints, it is easy to dismiss QoS provisioning in WSNs as implausible. However, there is one important difference between ad hoc networks and WSNs. WSNs are defined by applications they are deployed for. A large set of WSN applications, including security and surveillance, requires QoS guarantees from the network. Note that QoS-based applications were not integral parts of ad hoc networks and were proposed as additional applications that could run in parallel with non-QoS applications. Hence, QoS provisioning is a *requirement*, and not an optional feature, for WSNs.

Being multi-hop networks, WSNs potentially suffer under the same shortcomings and problems as ad hoc networks if similar QoS provisioning mechanisms are adopted. Furthermore, considering very limited resources in sensor nodes and the large scale of WSNs, per flow resource reservation-based approaches are especially ill-suited for WSNs. The two important differences between both network types suggests new directions in QoS provisioning: First, the mobility of WSNs is very limited when compared with ad hoc networks. Resource availability in WSNs fluctuates as a function of the offered load and not as a function of network connectivity over long periods of time. Therefore, communication decisions do not need to be updated very frequently. Second, the large scale of WSNs can easily be used as an advantage to eliminate explicit resource allocation. Distribution of communication responsibility over larger areas provides gains through diversity and allows local decisions to be made leading to end-to-end QoS guarantees.

QoS provisioning in WSNs is directly geared towards satisfying application requirements. In Figure 1, a generalized framework for information flow in a WSN is depicted. The main components of this framework are the sink, source locality, and relay nodes. The information flow starts with assignment of a particular task to sensor nodes. Upon information retrieval through sensors, source and other nodes nearby pre-process the information. The pre-processing may be simply forming data packets with raw data, data

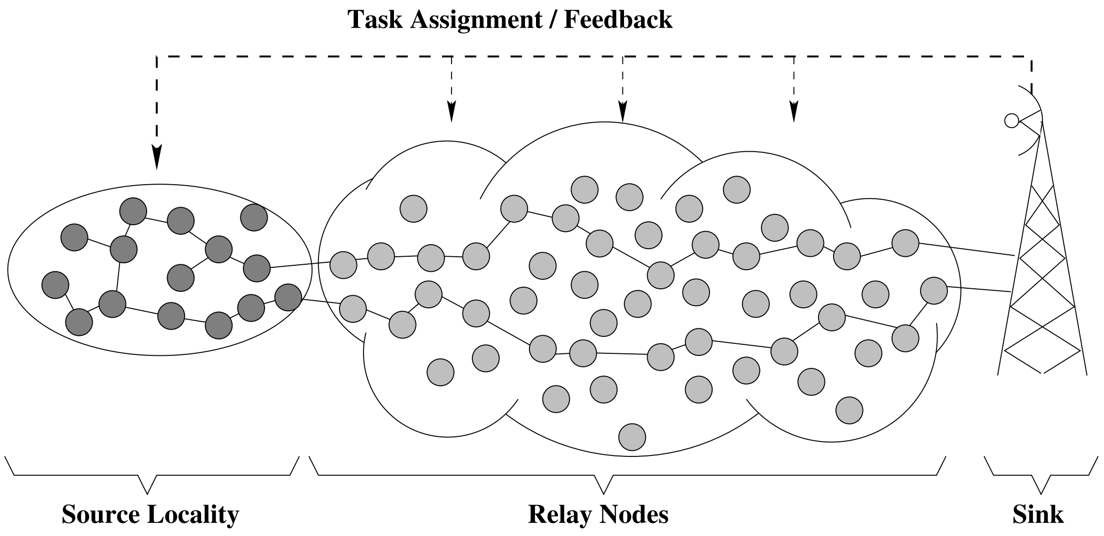


Fig. 1. Information Flow in a WSN

aggregation, or completely processing data and forming end results per application requirements. The information is then communicated via the relay nodes to the sink. In return, sink may optionally give feedback to the source locality and/or relay nodes. As will be presented in the next section, many of the QoS-based communication methods form an almost open loop where the source does not return any feedback to information sources or intermediate nodes.

III. QoS-Based MAC Protocols for WSNs

QoS provisioning in the MAC control layer mainly deals with the scheduling of packets on the wireless channel subject to local constraints. Since local constraints may change based on the needs of individual flows, the decisions are generally very dynamic and must be computed rather fast. The three solutions outlined below consider the requirements of real-time WSNs while scheduling medium access to contending nodes.

A. QoS-Aware Medium Access Control Protocol [27]

A QoS-Aware Medium Access Control protocol (Q-MAC) is presented in [27]. Q-MAC assumes an environment of multi-hop WSNs where nodes may generate packets with different priorities. The design objective of Q-MAC is to minimize energy consumption and provide QoS guarantees. Q-MAC is composed of intra-node and inter-node QoS scheduling mechanisms. The intra-node QoS scheduling scheme classifies outgoing packets according to their priorities, while the inter-node QoS scheduling solution handles channel access with the objective of minimizing energy consumption via reducing collision and idle listening.

The intra-node scheduling mechanism employs multiple First-In-First-Out (FIFO) queues with different priorities, among which an *instant queue* has the highest priority and its enqueued packets are always instantly served. The intra-node scheduling mechanism is outlined in Figure 2. Self-generated and relayed packets are classified to different queues with several QoS metrics, such as content importance and number of traveled hops. Data rate allocation between queues and serving packet selection are achieved through the MAX-MIN fairness algorithm [6] and the GPS algorithm [29], respectively.

Intra-Node_Scheduling:

```
1. WHILE TRUE
2.     IF new packet arrives
3.         Enqueued to different queues after classification
4.     WHILE instant queue is not empty
5.         Deliver the first packet in the instant queue
6.     IF there are queues not empty
7.         Select $q$ among these queues with the MAX-MIN and GPS algorithms
8.         Deliver the first packet in $q$
```

Fig. 2. The Intra-Node Scheduling Mechanism of Q-MAC

After a packet is scheduled for transmission, the inter-node scheduling mechanism, Power Conservation MACAW (PC-MACAW), is executed to achieve Loosely Prioritized Random Access (LPRA) between sensor nodes. In PC-MACAW, a successful transmission consists of two periods: The contention period and the packet transmission period. In the contention period, a node sends out RTS after waiting for a certain duration (contention time) and expects a CTS packet before accessing the channel. The contention time is randomly generated with a contention window size *CW*, where CW is determined by each node's transmission urgency including packet criticality, number of transmitted hops, residual energy, and queue's proportional load. After accessing the channel, the node enters the transmission period to send data packets and waits for an ACK packet. In case of collision, CW is doubled and the packet is retransmitted. When the difference between the current time and when the packet is generated exceeds a threshold, the packet is dropped. The PC-MACAW algorithm is outlined in Figure 3.

Q-MAC presents a combined effort of intra-node and inter-node QoS scheduling in WSNs. It is shown through simulations that Q-MAC provides the equivalent QoS while consuming less energy in comparison with an existing mechanism, S-MAC. However, complex scheduling mechanisms and relatively loosely defined QoS metrics stand out as shortcomings of this proposal.

B. Coloring-Based Real-Time Communication Scheduling [26]

The *Co*loring-Based Real-Time *Co*mmunication Scheduling (CoCo) solution is presented in [26]. CoCo is designed for multi-hop WSN environments that use IEEE 802.11 MAC protocol, where all communication is unicast. It is assumed that node locations are available at all times, and a central scheduler running CoCo is in charge of communication scheduling. CoCo aims to schedule real-time communication avoiding collisions and minimizing the overall packet transmission time.

In CoCo, a set of messages waiting for transmission at various sensors are modeled with a weighted, directed graph $G = (V, E)$, where a vertex denotes a sensor node, a directed edge from vertex v_i to v_j denotes a message to be sent from sensor v_i to v_j, and the weight of an edge denotes the transmission time. The communication problem is equivalent to assigning a color to each edge. Here, each color represents a set of simultaneous communication during disjoint time periods, and the weight of a color equals the maximum weight of the edges assigned with this color. CoCo aims to find edge color assignment such that (i) no adjacent edges share the same color, (ii) no two edges with the same color interfere with each other, and (iii) the overall weight of used colors is minimized.

```
PC-MACAW:
1. /*Enter the contention period*/
2. Backoff for a random number of time slots (ranging in [1,CW])
3. IF channel is not clear during the backoff time
4.     IF the dropping threshold is not reached
5.         GOTO Step 2 with doubled CW
6.     ELSE
7.         Drop the packet
8. ELSE
9.     Send RTS packet
10.    WAIT CTS packet
11.    IF the CTS packet is absent
12.        IF the dropping threshold is not reached
13.            GOTO Step 3 with doubled CW
14.        ELSE
15.            Drop the packet
16.    ELSE
17.        /*Enter the packet transmission period*/
18.        Transmit the packet
19.        WAIT ACK packet
20.        IF the ACK packet is absent
21.            IF the dropping threshold is not reached
22.                GOTO Step 2 with doubled CW
23.            ELSE
24.                Drop the packet
```

Fig. 3. The Inter-Node Scheduling Mechanism (PC-MACAW) of Q-MAC

Since the optimal coloring problem is NP-complete, a coloring heuristic is presented in [26]. First, the edges of the vertex with the maximum degree are assigned different colors. Once a color is assigned to an edge, it is removed from the palettes of all adjacent edges, and its weight is updated. Then, the following steps are repeated until all edges are colored: The edge with the smallest palette is chosen. A color from the available palette is assigned to the edge such that no other edge with that color interferes with the chosen edge. Then, the chosen color is removed from the palettes of all uncolored adjacent edges. Three heuristics are presented for selecting a color from an edge's palette: The *Random Color Selection Heuristic* randomly picks a color from the palette that does not cause interference. The *Least Used Color (LUC) Heuristic* chooses the color with the smallest number of colors. The *Minimal Weight Color (MWC) Heuristic* first checks whether there are colors in the palette whose weights are higher than the edge. If so, among these colors, the color with the smallest weight is selected. Otherwise, the color with the maximum weight is assigned from the palette.

CoCo aims to schedule a set of communication events with the minimum communication time in real-time WSNs. According to the simulations, MWC-based CoCo provides superior performance than the other two color selection heuristics, and its performance is close to the optimal solution. The central computation requirement limits the applicability of CoCo in large scale sensor networks.

C. Reliability Maintenance Through Activity Management [28]

In [28], an activity management mechanism (AMM) is proposed to maintain communication reliability in WSNs. WSNs are considered to work with underlying IEEE 802.15.4 protocol operating in beacon-enabled slotted CSMA/CA mode. Sensors are organized with a star topology, where a node must be admitted by a coordinator to participate in the network. The coordinator is aware of the number of nodes in the network, packet arrival rates, and the desired reliability R. Here, the reliability R is defined as the number of packets delivered to the coordinator per unit time. AMM aims to provide network reliability guarantee through sensor node activity management.

In AMM, the MAC layer exhaustively serves packets in First Come First Serve (FCFS) manner. A sensor goes to sleep only after all packets in its buffer are transmitted. During the sleep period, newly arriving packets are enqueued waiting for sensor wake up to be delivered. In case of queue overflow, the packet at the head of the queue is discarded. The sleep duration is geometrically distributed with the parameter P_{sleep}. A packet is transmitted starting with a random backoff countdown. After the countdown, two Clear Channel Assessments (CCA) are executed by listening to the channel to make sure it is idle. If both CCAs pass, the packet transmission starts and an ACK packet is expected. In case an ACK is not received, the transmission is repeated and the countdown exponent is increased. Through theoretical analysis of the throughput of the MAC mechanism above, the network control equation is derived where the network reliability R is a function of several MAC parameters, such as sensor sleep duration, transmission success probability, and basic backoff period. Given desired network reliability R, the coordinator calculates the parameter P_{sleep} and broadcasts it to all nodes to regulate their sleep times accordingly.

AMM provides network reliability thorough activity management and analysis of MAC parameters. It is shown through simulation that through proper activity management, the network reliability is robust against variation of network scale and packet rates. As the author point out, AMM is computationally intensive, and distributed activity management is left to future work.

IV. QoS-Based Routing Protocols for WSNs

QoS-based routing protocols for WSNs have been proposed in the literature mainly to support two kinds of performance bounds, namely, delay and reliability. The protocols outlined in this section are implemented primarily in network layer, and in some instances, in the MAC sublayer. These solutions also differ in the management of resources, where some rely on centralized computations while others utilized distributed methods. The common point in all these solutions is that all of them guarantee at least one performance metric to be satisfied in the network.

A. Sequential Assignment Routing [34]

On-demand multihop routing algorithms such as AODV and TORA eliminate table updates in high mobility scenarios. However, they introduce high energy cost during route setup phase. Power-aware routing finds minimum metric paths on two different metrics: Minimum energy per packet and Minimum cost per packet. The first metric produces substantial energy savings, but performance degradation due to link/node failure is not addressed. The second metric deals with failures by routing traffic away from low-energy nodes at the expense of high path maintenance cost. The Sequential Assignment Routing

(SAR) [34] algorithm uses the idea of multiple paths while taking parameters like energy resource, QoS on each path, and the priority of packets into consideration.

In SAR, a table-driven multipath approach is used to improve energy efficiency in a low mobility sensor network. The failure protection is addressed by having at least k-paths that have no common branches between a node and a sink. This is called a k-disjoint structure. However, the disjoint property creates strong coupling between routing tables, rendering localized recovery schemes ineffective. To reduce this effect, the disjoint requirement is relaxed outside the 1-hop neighborhood of the sink. Furthermore, localized path restoration procedures are used to lower energy cost in failure recovery. Multiple paths from each node to a sink are created by building multiple trees each rooted at the 1-hop neighborhood of the sink. Each node uses two parameters to create routing paths:

- Energy resource which is estimated by maximum number of packets that can be routed without energy depletion, assuming that the node has exclusive use of the path.
- Additive QoS metrics where higher metric implies lower QoS.

Path selection is made by nodes that generate packets if no topology change occurs while packets are being routed to their destinations. The energy cost and delay of links are considered as additive QoS metrics. Packet priorities are used in a way that packets with higher priorities use paths with lower latency. In short, for each packet, a weighted QoS metric is computed as the product of a weight coefficient (the priority of the packet) and the additive QoS metric. Hence, QoS is provided to each packet relative to its priority level, where higher QoS is given to higher priority level packets. The SAR algorithm minimizes the average QoS metric throughout the lifetime of the network.

Periodic metric updates triggered at the sink node are used to account for possible changes in the QoS on individual paths and the changes in energy resources. Simulations show that SAR performs better than a minimum metric algorithm that lowers energy consumption without considering packet priorities. Furthermore, failure recovery is handled by local handshakes between upstream and downstream neighbors in paths. SAR algorithm addresses low mobility networks and routes are established at packet sources considering link costs and energy resources as a QoS parameter. Packet priorities are taken into account to relay high priority packets to popular paths in terms of latency. However, the scheme requires resource-related topology information at packet sources which requires frequent parameter updates by a common sink. This incurs high overhead in WSNs with moderate or high mobility, and in WSNs carrying high data rates.

B. Energy-Aware Routing in Mobile and Wireless Ad Hoc Networks [9]

Before focusing on energy-aware routing protocols for WSNs, it is worth focusing on energy-saving routing protocol design for wireless ad hoc networks. These solutions are also directly applicable to WSNs with limited number of nodes and where a number of potentially mobile, high-capability nodes need to communicate possibly over sensor nodes. A good example of modifying existing routing protocols to be energy-aware has been presented in [9]. In this paper, two reactive ad hoc routing protocols, namely DSR [24] and TORA [30], are modified to deliver QoS by introducing energy-awareness.

DSR and TORA protocols involve a ”route discovery phase” initiated when a mobile source node needs to send data packets to the destination node, but does not have an active and valid route to it.

Route discovery is performed using the control packets called route request packets (RREQ). The source node broadcasts RREQs and waits for a control packet called the route reply packet (RREP). Using the received RREP, the source node updates its routing table which is used to keep track of active routes to individual destination nodes in the network. Intermediate nodes, which forward RREQ and RREP packets between source and destination nodes, also use RREP packets to update their routing tables. Although these protocols are effective and robust, the broadcast of RREQ packets leads to unnecessary packet transmissions and inefficient use of limited energy resources.

In [9], two modified protocols, EDSR and ETORA, based on the existing DSR and TORA protocols are introduced, respectively. Both EDSR and ETORA involve an additional RREQ forwarding mechanism that does not exist in DSR and TORA. In an intermediate node, this mechanism considers the current energy level of the node, the energy level of the previous sender node, and the distance to the source node when making routing decisions.

Distance estimation is accomplished using time stamps in RREQ packets. When a node transmits an RREQ packet, it records the time of transmission in the RREQ. Intermediate nodes that receive this RREQ calculate the time difference between transmission and the reception time of the RREQ to estimate their distances to the sender node. Besides transmission times, energy levels of the nodes right before RREQ transmission are also recorded in RREQs.

The RREQ forwarding decision in [9] is based on cutoff circles that are placed around each node in a network. Upon the reception of an RREQ packet, an intermediate node calculates the diameter of its ”cutoff” circle using its energy level, the energy level of the previous node sending this RREQ packet, and its distance to the previous node calculated by the time stamp in the RREQ packet. The receiving node simply drops the packet, hence does not forward it, if its cutoff circle encircles the previous node. In [9], ETORA and EDSR are shown to outperform TORA and DSR, respectively, in terms of overall network throughput, the average number of data packets received at destinations, average data transmission delay, and energy consumption. The pseudo-code of the proposed RREQ forwarding algorithm is given in Figure 4.

C. Energy-Aware QoS Routing Protocol for WSNs [1]

The information delivery in video sensor networks requires end-to-end delay guarantees. The QoS-based routing protocol proposed in [1] aims to sustain paths that can guarantee such delays for real-time traffic while supporting non-real-time (best-effort) data flows, as well. The network architecture assumed in this proposal involves a hierarchical organization: Sensor nodes are grouped into clusters, formed based on criteria such as communication range, number and type of sensors, and geographical location. Clusters are assumed to be controlled by a single command node and have their own gateway nodes, which act as cluster heads. Sensors in a cluster receive commands from and send readings to the cluster gateway node. Gateway nodes of different clusters are able to communicate over long-haul communication links. All sensors are assumed to be stationary. In addition to non-real-time data generated in the network, the network also tries to recognize and track targets in individual clusters using images and video feeds.

In this study, the aim is to find paths within a particular cluster under real-time constraints, without explicitly addressing communication among gateways. Given a cluster of sensor nodes, paths are computed

```
Forward_RREQ:
    $tp$: Time of transmission by the previous hop
    $tr$: Time of reception by the intermediate node (IN)
    $Ep$: Energy level of previous node during RREQ transmission
    $Er$: Energy level of this IN when RREQ is received
    $d_c$: Diameter of the cutoff circle of this IN
    $l$: Distance to the previous hop
    $c$: Speed of light

1. Calculate time difference $\Delta_t := tr - tp$
2. Estimate distance $l := \Delta_t \times c$
3. Determine diameter of cutoff circle $d_c = 0.4 \cdot Ep + 0.4 \cdot Er + 0.2l$
4. IF $d_c \leq l$
5.     Drop RREQ packet
6. ELSE
7.     Forward RREQ packet
```

Fig. 4. Energy Efficient RREQ Forwarding Algorithm

centrally by gateways. To this end, a cost is associated with each link. A link cost is a weighted sum of physical length of the link, residual energy of the sender, expected lifetime of the sender assuming current power consumption rate, and the estimated error rate of the link. Using these link costs, the gateway computes k-shortest paths between individual nodes and the gateway itself. However, this computation is not sufficient to satisfy the delay constraints of flows. Based on pre-computed k-paths between sources and the gateway, the gateway also computes the expected delay on every link. Assuming perfect knowledge about the demand of data sources and a fixed ratio r of resources used for real-time flows, a queuing model is formed. This queuing model is used to compute the delay between data sources and the gateway. The path computation algorithm aims to find a common ratio r such that end-to-end delay requirements of all nodes are satisfied for at least one of the k-paths associated with each data source. Once computed, the resource ratio r is broadcast to all nodes along with the selected paths to route information in the network.

This work has an interesting approach to the QoS provisioning in the network where both real-time as well as non-real-time data flows are considered in the network. The proposed solution is expected to perform well when the size of the cluster is small. However, the link cost function helps minimizing the energy consumption for appropriate weights for different components of the cost. The authors do not comment on the selection of the weight parameters. Furthermore, the assumption of knowing the data rates of individual sensor nodes as well as the up-to-date state information of relay nodes at a central location is far from being realistic. While this algorithm is a nice attempt to solve a complex problem, it only does so at the expense of limiting simplifications and assumptions of cluster-wide state information.

D. Energy-Aware Data-Centric Routing Algorithm for WSNs [8]

An energy aware and data-centric routing algorithm (EAD) for WSNs is proposed in [8]. The algorithm aims to achieve two performance improving goals: (i) Elimination of redundant data by in-network processing and (ii) Minimization of overall network energy consumption by using a virtual backbone tree for data forwarding. The EAD algorithm is mainly focused on the construction and maintenance of the forwarding backbone tree rooted at a single data sink with maximal number leaf sensor nodes. The network operation is composed of two major phases, namely the *initialization* and the *data transmit* phases. Initialization and data transmit phases together form a “round”. The construction and following updates of the backbone tree are performed during the initialization phase.

The construction of the forwarding tree is based on the following idea: The dominant part of the energy consumption in a sensor node is due to data transmission and reception. To minimize the overall network energy consumption, some nodes should turn their radios off (leaf nodes) while others should remain relaying packets (non-leaf nodes). Hence, to achieve minimal energy consumption, the focus of attention is on the maximization of the leaf nodes. Since this is an NP-complete problem, the EAD algorithm heuristically attempts to achieve this goal. The algorithm periodically updates the distribution of the leaf and non-leaf nodes during the initialization phases by considering a sensor node as a state machine.

The choice of being a non-leaf node depends on two mechanisms, namely neighboring broadcast scheduling and distributed competition among neighbors. These mechanisms ensure that sensors with higher residual power have higher chance to become a non-leaf node, hence conserving the local energy which eventually leads to reduction in overall network energy consumption. Leaf and non-leaf nodes change their states upon the reception of messages from neighbors indicating their parents, energy levels, and distance to the sink node. Nodes sense the channel before transmitting these messages and also have waiting periods to avoid unnecessary local state changes.

In EAD, data relaying and in-network data processing tasks are performed in non-leaf nodes. At each sensor, the local raw data is combined with partially processed data delivered from sensors that are farther away from the sink. (The sensor nodes keep records of their parent nodes and their child nodes, if any, through which they determine their distance to the sink.) Non-leaf nodes summarize and forward the aggregated data to their parents in the tree. To reduce the execution time of EAD processing, a topology-based algorithm is used which preprocesses the network topology to ensure that all sensors are are spanned by the EAD tree even though a subset of sensors participate in EAD execution. The outline of the two algorithms used in EAD are given in Figures 5 and 6.

The performance of the EAD algorithm is compared with the performance of a simplified AODV without sensor mobility, and LEACH algorithms comparing total number of active nodes, UDP packet throughput, and energy expenditure. The presented results illustrate that EAD outperforms the other two algorithms with respect to these performance metrics. Furthermore, it is shown that there is a trade-off between system lifetime and system throughput when shorter or longer EAD refresh periods are chosen. Small refresh intervals enable better throughput but require more energy. Moreover, it is concluded that there is no trade-off between the initialization and data transmit phases of the EAD algorithm.

E. Reliable Information Forwarding using Multiple Paths [15]

Data dissemination protocols which are not adaptive to channel error rates and do not support information awareness either spend excessive amount of resources or fail to deliver important information with sufficient reliability. Reliable Information Forwarding using Multiple Paths (ReInForM) [15] is a protocol for WSNs to support information awareness, such that the reliability of data transfer depends on

Receive_Control_Packet:

n:	Current node
$n.nodeId$:	Unique ID attribute of n
$n.EAD\_type$:	Type attribute of n (0: undetermined, 1: leaf, 2: non-leaf)
$n.EAD\_previous\_type$:	Previous type attribute of n
$n.EAD\_level$:	Current level attribute of n
$n.EAD\_parent$:	Distance attribute of n to the previous hop
$n.EAD\_has\_child$:	Boolean attribute of n indicating if it has children
sink:	
P:	Received packet
$P.ead\_type$:	Type of the source of P
$P.ead\_level$:	Level of the source of P
$P.ead\_parent$:	Parent node of the source of P
$P.source\_add$:	Source address of P

```
1.  $n.EAD\_previous\_type := n.EAD\_type$
2.  **IF** $n.EAD\_type == 0$
3.      **IF** $P.ead\_type == 2$
4.          $n.EAD\_type := 1$
5.          $n.EAD\_level := P.ead\_level + 1$
6.          $n.EAD\_parent := P.source\_addr$
7.      **ELSE IF** $P.ead\_type == 1$
8.          $n.EAD\_type := 2$
9.          $n.EAD\_level := P.ead\_level + 1$
10.         $n.EAD\_parent := P.source\_addr$
11.     Call *finalEADStatusUpdate* function
12.     Send control packet to 1-hop neighbors
13. **ELSE IF** $n.EAD\_type == 1$
14.     **IF** $P.ead\_parent == n.nodeId$
15.         $n.EAD\_type := 2$
16.         Call *finalEADStatusUpdate* function
17.         Send control packet to 1-hop neighbors
18. **ELSE IF** $n.EAD\_type == 2$
19.     **IF** $P.ead\_type == 2$ **AND** $P.ead\_parent == n.nodeId$
20.         $n.EAD\_type := 2$
21.         Call *finalEADStatusUpdate* function
22.         Send control packet to 1-hop neighbors
23. **IF** $P.ead\_parent == nodeId$
24.     $n.EAD\_has\_child := TRUE$
25.     $n.EAD\_type := 2$
```

Fig. 5. Control Packet Reception Procedure used in EAD Algorithm

the information content despite the presence of significant channel errors. To define the desired reliability levels, ReInForM assigns different priority levels to data packets. Depending on the priority level, multiple copies of the data packets are delivered along multiple paths. Hence, ReInForM relies heavily on the existence of multiple paths between a source and a destination, which is generally available in large scale WSNs. The simulations investigating the existence and number of edge-disjoint paths show that a network slightly denser than a minimally connected graph is sufficient to have as many edge-disjoint paths as the average node degree. The deviation in the number of hops of these paths is found to be less than two hops, which suggests that the paths have nearly identical lengths. Hence, data delivery on these paths has

finalEADStatusUpdate:

```
$n$: Current node
$P$: Received packet

1. IF $P.ead\_parent == nodeId$
2.     $n.EAD\_has\_child := TRUE$
3. IF $n.EAD\_has\_child == TRUE$
4.     $n.EAD\_type := 2$
5. ELSE IF $n.EAD\_previous\_type != 0$ AND $n.EAD\_has\_child == FALSE$
6.     $n.EAD\_type := 1$
```

Fig. 6. EAD Status Update Procedure used in EAD Algorithm

comparable latency and efficient load balancing among multiple paths is possible.

Under ReInForM, the source node of a packet determines the importance of the information in the packet and decides on a reliability level (r_s). Using the local channel error information (e_s) and the hop distance to the sink (h_s), the source computes the number of paths, $P(h_s, r_s, e_s)$, required to deliver the packet at the chosen reliability level. The neighbors of the source is divided into 3 subsets, ${H_s}^-$,${H_s}^0$, and ${H_s}^+$, designating the neighborhoods at distances of ${h_s}^-$, ${h_s}^0$, and ${h_s}^+$ hops to the sink, where ${h_s}^- < {h_s}^0 < {h_s}^+$ and $h_s = {h_s}^0$ is the hop distance of the source to the sink. The chosen total number of paths that the source is expected to create, P, is divided into these three sets of neighborhood.

A random node in ${H_s}^-$ is chosen to be the default node, which always forwards packets. This ensures to have at least one path towards the sink. Other nodes in ${H_s}^-$, using their own local channel error rate e, hop distance to sink h, and reliability r, compute their own P value. If this P value is larger than 1, then the node is chosen to be a forwarding node. If the value is less than 1, then the probability that the node is a forwarding node is simply this local P value. Eventually, a number of forwarding nodes are chosen in set ${H_s}^-$. If there are still more paths to be established (meaning that the number of paths over the set ${H_s}^-$ is less than the total number of paths), then additional paths are created by the nodes in the set ${H_s}^0$ in the same way. Paths over the set ${H_s}^+$ are created only if there are still more paths needed besides the ones created by ${H_s}^-$ and ${H_s}^0$.

The nodes that decide not to forward a packet simply drop it. Packets carry minimal state information to aid the forwarding decisions. The dynamic local states containing hop distance to sink, reliability, and channel error rate are updated regularly at each forwarding node. After receiving a packet from the source and updating the dynamic states, the node effectively becomes a source. Using the local state information, the node uses the same procedure to compute the path values for its own neighbors and the process continues.

ReInForM is one of the leading examples of multipath routing protocols for WSNs. The gains attained through local multipath forwarding mechanisms lend predictability to applications in terms of reliability. While load balancing is argued to be a natural result of the protocol, the study does not present any the analysis of this issue. Furthermore, not all alternative edge-disjoint paths are of equal length, potentially leading to out of order delivery of packets and unpredictable delays.

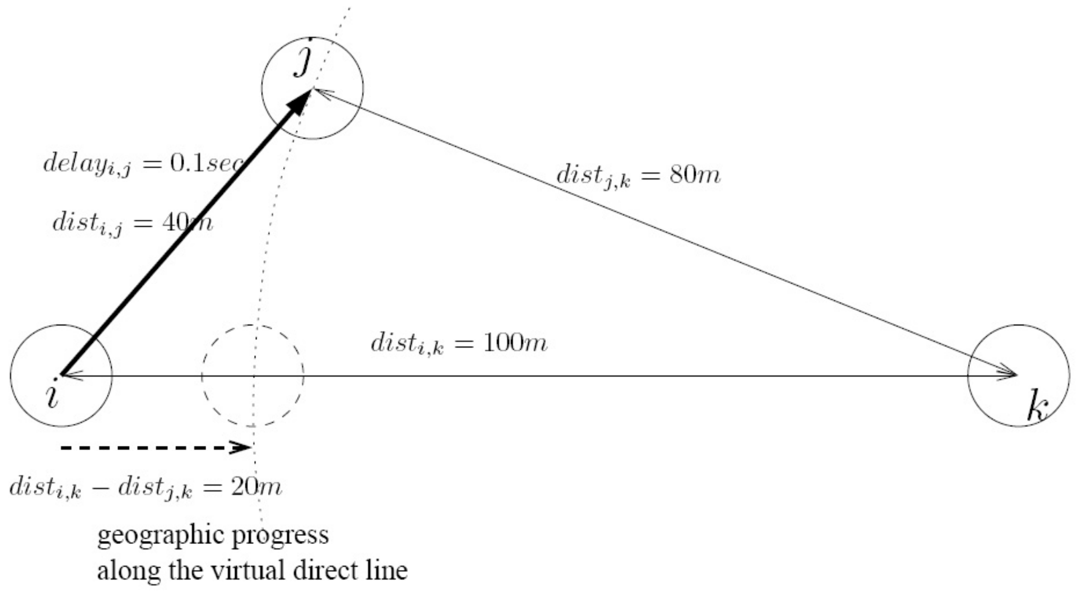


Fig. 7. Packet Progress Speed Calculation under SPEED Protocol

F. SPEED Protocol [23]

In large scale WSNs, the availability of state information of individual nodes cannot always be assumed. Therefore, local decision based solutions prove to be more flexible as well as feasible for such large scale networks. SPEED protocol [23] is designed to provide soft end-to-end deadline guarantees for real-time packets in WSN. It uses a geographic forwarding mechanism such that each packet can be routed without global topology information. Thus, it scales well in large sensor networks. More importantly, it ensures a network wide speed of packet delivery for real-time guarantees. The network is assumed to be composed of nodes that have location information. Since the protocol is based on a geographic routing algorithm, nodes are also assumed to gather information about their neighbors' locations.

The concept of soft real-time guarantees in the context of the SPEED protocol refers to the fact that packets travel to their destination at a give propagation speed. For this, each node maintains information about neighboring nodes such as geographic distance and average delay to each neighbor. Using the distance and delay, each node evaluates the packet progress speed of each neighbor node for a packet sent to a specific destination. An example of packet progress speed calculation is shown in Figure 7. Let a packet in node i be destined to k, which is $100m$ away. Let i forward the packet through a neighbor j, which is $80m$ away from k. If this forwarding takes, on the average, $0.1s$, then the packet progresses to its destination k at $\frac{20m}{0.1s} = 200m/s$. Under SPEED protocol, a packet is forwarded through a neighboring node if and only if the progress speed through that neighbor is higher than the specified lower-bound speed *SetSpeed*.

If each node can find a neighbor that can progress a packet with a speed higher than *SetSpeed*, *SetSpeed* can be guaranteed in the entire network. However, if the load carried in the network is too high, uniform speed guarantees cannot be provided in the network. When a node cannot find any neighbor node whose speed is higher than *SetSpeed*, it probabilistically drops packets to regulate the workload such that at least one neighbor node with a speed higher than *SetSpeed* exists at all times. At the same time, the node sends a back-pressure packet to the previous nodes to prevent them from forwarding any further packets through this congested area. Hence, packet delays can be upper-bounded at the expense of packet losses in the network to sustain network-wide packet progress guarantees. The algorithms used in the SPEED protocol are given in Figures 8 and 9.

With the SPEED protocol, a uniform packet progress speed is guaranteed in the entire network. Furthermore, the protocol relies only on local information augmented with limited scope feedback, which improves its scalability. However, the SPEED protocol provides only one network-wide speed, which is not suitable for differentiating various flows with different deadlines. Furthermore, it does not provide any guarantees in packet delivery: The fraction of packets lost or dropped in the network cannot be know ahead of time. This, in turn, renders the SPEED protocol not entirely suitable for real-time applications for WSNs.

G. MMSPEED Protocol [17]

The two important shortcomings of the SPEED protocol [23] are the support of a single propagation level and the lack of support for ensuring end-to-end reliability. The MMSPEED protocol [17] addresses these two shortcomings. This is achieved by creating multiple logical layers in the same physical network. In Figure 10(a), two logical delay layers that provide two different packet propagation speeds on the same physical network are depicted. To provide delay guarantees, the idea of providing network-wide speed guarantees is adopted [23] for each logical layer. The speed layers are isolated from each other through prioritization in queuing and channel access. Figure 10(b) shows how different reliability levels can be provided in the same network. To guarantee different reliability levels, packets are routed over multiple paths based on the requirements contained in headers and on local statistics on packet loss.

Local forwarding decisions are made considering local statistics and required speed and reliability levels contained in every packet's header. Let a packet x be generated by the source s. The source node s calculates the required speed $S^{req}(x)$ for x so that x reaches its destination d by its deadline $D(x)$, i.e., $S^{req}(x) = \frac{|s,d|}{D(x)}$, where $\mid s, d \mid$ is the distance between s and d. The source node s includes the required speed $S^{req}(x)$ in x's header. First, let us consider the delay QoS provisioning in a network which supports L layers of propagation speed $S_1, \cdots, S_L$. At the source, s selects the minimum speed layer l larger than $S^{req}(x)$, i.e., $S_l = \min_{j=1}^{L}\{S_j \mid S_j \geq S^{req(x)}\}$. Then x is forwarded to one of the neighbor nodes i that has a propagation speed of $S_{s,i}^{d} = \frac{|s,d|-|i,d|}{d_{s,i}}$ with respect to d, where $d_{s,i}$ is the delay estimate from i to s, and $S_{s,i}^{d} \geq S_l$. As the packet is forwarded in the network, it may be propagated slower than anticipated due to random node placement and lack of coordination between distant nodes. If an intermediate node finds that a packet cannot reach its destination at a speed layer, the packet is pushed to a higher speed layer with a new required speed S^{req}. Hence, errors in local decisions are compensated as packets traverse the network.

To provide reliability in reaching the destination, the packet loss rates to neighbors are monitored and the total number of hops to the destination is estimated. The source node s includes the reliability requirement $R^{req}(x)$ of packet x in the header. Consider x being forwarded by node i. All nodes including i keep packet loss statistics for their neighbors j, indicated by $e_{i,j}$. The end-to-end reachability estimate $R_{i,j}^{d}$ from node i to destination d over neighboring node j is calculated as $R_{i,j}^{d} = (1-e_{i,j})^{K}$, where $K = \left\lceil \frac{|i,d|}{|i,d|-|j,d|} \right\rceil$ is the

Forward_Packet:

i:	Current node
D:	Destination node
$d(i, D)$:	Destination of node i to D
NS_i:	Neighborhood set of node i
FS_i:	Set of all nodes in NS_i closer to D than i
$HopDelay(i, j)$:	Estimated delay from i to j
$Speed(i, j, D)$:	Estimated speed from i to j
p:	Packet being processed
BPP:	Back-pressure packet
US_i:	Upstream Node Set of node i
$S_{setpoint}$:	Speed threshold
$FS_{i,set1}$:	Nodes in FS_i with sufficient speed
$FS_{i,set2}$:	Nodes in FS_i with insufficient speed
e_n:	Miss ratio of neighbor n
RR:	Relay ratio
K:	Proportionality gain

```
1.  $FS_i := FS_{i,set1} := FS_{i,set2} := \emptyset$
2.   FOR all $k$ in $NS_i$
3.      $L_{next} := d(i, D) - d(k, D)$
4.      IF $L_{next} > 0$
5.         Add $k$ to $FS_i$
6.         $Speed(i, k, D) := (L - L_{next})/HopDelay(i, k)$
7.         IF $Speed(i, k, D) > S_{setpoint}$
8.            Add $k$ to $FS_{i,set1}$
9.         ELSE
10.           Add $k$ to $FS_{i,set2}$
11. IF $FS_i == \emptyset$
12.     Drop $p$
13.     FOR all $u$ in $US_i$
14.        Send $BPP$ to $u$
15. ELSE IF $FS_{i,set1}! = \emptyset$
16.     Choose $k$ in $FS_{i,set1}$ with $\max(Speed(i, k, D))$
17.     Forward $p$ to $k$
18. ELSE
19.     $R := 1 - K \times mean(e_n)$
20.     Generate random number $RN$ in $[0, 1]$
21.     IF $RR < RN$
22.        Drop $p$
23.        FOR all $u$ in $US_i$
24.           Send $BPP$ to $u$
25.     ELSE
26.        Choose $k$ in $FS_i$ with $\max(Speed(i, k, D))$
27.        Forward $p$ to $k$
```

Fig. 8. Stateless Non-geographic Forwarding Algorithm of the SPEED Protocol

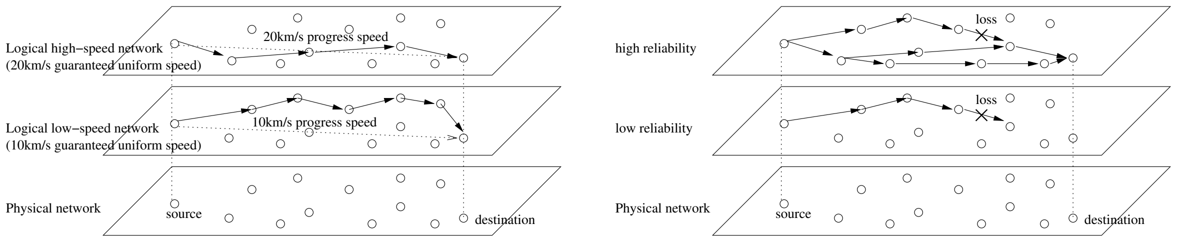
estimate of hop distance from i to d. After determining its neighbors that can sustain the required speed for packet x, node i chooses a subset of neighbors $j_1, \cdots, j_m$ such that $1 - \prod_{n=1}^{m}(1 - R_{i,j_n}^{d}) \geq R^{req}(x)$. In other words, we try to make sure that the probability of one copy of x to reach d is not smaller than the original required reliability level. At the same time, we make sure that the packet is propagated at the required speed. In each of the copies sent to these neighbors j_n, the required reliability of packet x is

Process_BPP:

i:	Current node (recipient of the BPP)
j:	Node with congested downstream neighbors(sender of the BPP)
D:	Destination node
$SenttoDelay_j$:	Delay of downstream node j
$AvgSenttoDelay$:	Average of $SenttoDelay$s of nodes in FS_j

```
1. IF $j \notin FS_i$
2.     Drop BPP
3. ELSE
4.     Adjust $SenttoDelay_j$ with $AvgSenttoDelay$
5.     IF $i$ is congested for all $k$ in $FS_i$
6.         Send BPP to all upstream nodes of $i$
```

Fig. 9. Back-Pressure Packet Handling Algorithm of the SPEED Protocol



(a) Delay QoS Domain

(b) Reliability QoS Domain

Fig. 10. Two QoS Domains and Corresponding Layers Implemented in the Same Network

updated as $R^{req}(x) = 1 - e_{i,j_n}$. Hence, as the number of paths x is relayed over increases, the individual reliability levels decrease while preserving the total reliability level. Note that the probability of discarding a packet to sustain a speed level increases as the required reliability for the packet decreases.

At this point, we would like to emphasize the interactions between the network layer and the MAC layer. The network layer makes decisions about the selection of forwarding nodes for each packet and maintains a prioritized queue for different speed layers. On the other hand, the MAC layer implements mechanisms for prioritized access to the channel according to speed levels and implements a multicast mechanism. The network layer makes its forwarding decisions based on the information it obtains from the MAC layer. The MAC layer monitors the delay and packet loss probabilities for each neighbor.

The MMSPEED protocol provides delay as well as reliability guarantees to real-time flows. With its dual objective nature, MMSPEED stands out as a multi-faceted protocol that cuts across MAC and network layers. These features come at the price of higher complexity in monitoring the network as well as storage requirements. Hence, added storage and processing requirements may potentially increase the cost of individual nodes in the sensor network.

H. Dynamic Delay-Constrained Minimum Energy Dissemination Protocol [25]

Some real-time applications for WSNs may require multiple sinks to obtain sensory data from a single source. At the same time, the end-to-end data delivery delay between a data source and sinks is required to be upper bounded. The Dynamic Delay-Constrained Minimum Energy Dissemination (DEED) protocol [25] aims to form and maintain multicast trees in the WSN that minimize energy consumption while guaranteeing end-to-end delays. Since the number and locations of the sinks may not be available, it is required to dynamically adapt the data dissemination tree upon the arrival of new sinks and leaving of the existing ones. A single source node in the WSN sends data and arbitrarily located mobile/stationary sinks request this data. Sensor nodes are assumed to be aware of their own geographic locations and their immediate neighbors. Moreover, each sink has an upper bound for end-to-end message delivery delay (UBED) from the source.

The dissemination tree (d-tree) is rooted at the data source node. New sensors are added to the tree as new sinks request data from the source and register themselves with the tree. Hence, the construction of the d-tree starts from the single source node, and continues as new sinks arrive. The DEED scheme deals with this construction while minimizing the total energy consumption and meeting UBED requirements. The structure of the d-tree has two main parts, namely, the static part and the mobile part. The static part consists of the source, the sensors (relays) and the multihop edges between the relays. The sensors that connect sinks to the d-tree are called *access relays (ARs)*. The mobile part of the d-tree is simply the set of one-to-one connections between sinks and their corresponding access relays (AR).

The end-to-end delay in the DEED protocol (which is upper bounded by the UBED D_m for each destination m) is composed of the end-to-AR delay upper bounded by P_m and the delay between a sink and its access relay, upper bounded by δ_m, where $\delta_m + P_m = D_m$. The delay through multiple hops is the sum of queuing, transmission, propagation, MAC, and retransmission delays. However, since the queuing, MAC, and retransmission delaya are unpredictable, DEED introduces a new parameter, the *average delay per distance* q, which is obtained through tests like ping applications and then given to sensor nodes. Furthermore, since the geometric distance of multihop edges are nearly proportional to hop count in a sensor network, geometric distance is used as a measure of delay along with the parameter q. The end-to-end delay is computed as the sum of the edge delays along an end-to-end path.

The tree-update procedure when a new sink joins is also the mechanism of d-tree construction. The packets for constructing the d-tree are forwarded by greedy forwarding, whereas data packets are broadcast and only the nodes that cache the addresses of the senders receive the packets. The procedure for d-tree construction is as follows: When a sink m wants to join the d-tree, m locates its nearest neighbor a_m and chooses a_m as its access relay. Then, the sink m sends a JOIN query over a_m to the source. Upon the reception of the JOIN query, the source subscribes the sink and its access relay. This procedure is called the *subscription phase*. Then, the source initiates the *gate-relay search* procedure. A gate relay is the relay in the existing tree where a separate branch of the tree is created to reach the access relay. The gate relay is searched recursively over the relays of the d-tree starting from the source node. Let $r[i]$ be the relay that is checked to be a candidate gate relay, where $r[0]$ is the source sensor. Let H be the union of the set of children of the relay node $r[i]$ in the d-tree and the relay node $r[i]$. Furthermore, let s_i be the delay from the source to $r[i]$. The objective in the gate relay search is to minimize $d(h, a_m)$, which is the distance between the access relay and a relay node in H. Moreover, the delay between the source and the access relay over this gate relay should be lower than the end-to-AR delay constraint P_m of the sink as follows:

$$(s_i + q[d(r[i], h) + d(h, a_m)]) < P_m \tag{1}$$

Here, h is the element of H with minimum $d(h, a_m)$. If h is found to be the relay node itself (such that $d(r[i], a_m)$ is minimum $d(h, a_m))$, then $r[i]$ becomes the gate relay. Otherwise, node h with minimum $d(h, a_m)$ is assigned to be the next relay node $r[i + 1]$ to run the same algorithm recursively and s_{i+1} is updated accordingly.

The third step of the three construction aims to locally adjust the tree around the gate relay (hence form a branch leading to the access relay) to produce an optimum dissemination tree from source to the destination. A junction node J in the neighborhood of g is searched such that the total of the distances from gate relay g, its closest child c, and the access relay a_m to the junction J is minimized. J should also satisfy the delay constraints of the sink m, and all the sinks that node c has as its descendants. If the delay constraints are not satisfied for all neighbors of g, then a_m becomes a direct child of node g. Otherwise, the chosen junction J becomes a relay of the new branch emerging from g towards a_m and J recursively runs the same algorithm until the delay conditions are not met for a junction's all neighbors. At this point all the relay nodes on the new branch are determined.

If the sink is mobile, the path leading to the sink and the distance between the sink and the access relay change. Hence, the sink notifies its access relay (AR) of its latest nearest neighbor node in order to continue to communicate with the AR. If the sink-to-AR distance is increased too much and the sink-AR delay constraint is violated, then the sink needs to select another access relay.

DEED is an efficient algorithm to minimize energy consumption and meet delay requirements. However, it does not propose service differentiation among flows towards different sinks. Furthermore, decrease of reliability in data delivery due to possible link errors in wireless channels is not addressed. Additional mechanisms to address path changes due to link errors and congestion are needed. DEED adjusts the delay/distance parameter to handle congestion but this does not guarantee avoiding highly congested local spots. Furthermore, these additional mechanisms must be local and should not alter the overall tree structure.

I. QoS for Data Relaying in Hierarchical WSNs [5]

This study presented in [5] addresses the selection of one or more routes from sensors to a base station. The routes are chosen such as to satisfy the delay requirements. In this study, large heterogeneous WSNs are considered which are organized in a three tiers of hierarchy: A base station (BS), relay nodes (RN) and finally sensor nodes acting as data sources. Relay nodes are placed such that connectivity is maintained. Additional relay nodes are placed to improve energy consumption and reduce interference. However, increasing the number of relay nodes increases the end-to-end delay. Furthermore, relay nodes that are closer to the base station consume more energy compared to others. To address these issues, a hybrid approach is proposed that introduces relay gateways (RG) that receive data from relay nodes and send them directly (in a single hop) to the base station. RGs are pre-deployed in the network and are stationary.

The routing decisions at RN-RG and RN-RN communication level consider the system lifetime as a constraint. System lifetime is defined as the time until at least one RN or RG depletes its energy supply.

The lifetime of a node is modeled by the maximum amount of traffic it can handle, which is called the node capacity. The routing from sensors to relay nodes is assumed to be handled by low level protocols. Hence, this protocol only deals with efficient routing of data among relay nodes by delivering it to one of the RGs while meeting the delay requirements. Two different cases of selecting a relay path are proposed: Multi-Path Relaying with Delay Constraints (MPD) and Unconstrained Multi-Path Relaying (MP) Two different algorithms which are both centralized and optimal are proposed to solve these problems.

In the MP problem, the aim is to minimize the end-to-end communication cost while meeting the capacity constraints of RNs and RGs. However, this problem does not deal with end-to-end delays. The WSN is modeled as a directed graph composed of a set of vertices (composed of RNs and RGs) and a set of arcs which represent the edges between vertices. The cost c of sending a packet over an arc is defined as a function of individual arcs. Furthermore, the flow of data over an arc is defined as a separate function x, which is used to model the routing decisions of every RN. Hence the total cost of communication over the selected arcs from source RN to RG, $\sum x(a)c(a)$, is minimized, where a is an arc.

The capacity of a node i is represented as $\gamma(i)$, whereas the total amount of data forwarded to a relay from sensors is $\beta(i)$, which is called the demand of i. The set of all arcs entering a node i is $\delta(i-)$ and the set of all arcs leaving i is $\delta(i+)$. The minimization of total cost is subject to $x(\delta(i+)) - x(\delta(i-)) = \beta(i)$, meaning that the demand of a relay node i is equal to the net flow into i, i.e., $x(\delta(i-)) - t\beta(i) \leq \gamma(i)$, where it is assumed that transmission requires $t/(1-t)$ more energy than reception.

The MP problem is modeled as a transshipment problem, which is a classical problem in operations research. The transshipment problem can be solved in strongly polynomial time. In MPD problem, apart from minimizing the total cost and meeting capacity constraints, the total number of intermediary nodes on a path should not be larger than a given value to limit the total delay. Hence, packets that originate at different RNs should be distinguished. The delay constraint can be formulated either using flow functions or using feasible paths. In the former, relay nodes increment a hop-count index which is assigned to individual flows. In the latter, the set of all feasible directed paths $P(r, k)$ originated from a RN r and ending at a RG k is defined. The paths with lengths smaller than a threshold are chosen. However, the demand of every node should be met and the capacity of all nodes should be observed. The MPD with feasible paths formulation can be solved optimally using a linear program. The advantage of feasible path formulation over the flow function formulation is that the number of constraints is linear in the size of the graph. Column generation, which is an implementation of the simplex algorithm for solving linear programs, is used to find a solution to the MPD problem.

This work is based on a graph representation of a WSN, which is used to find QoS-based paths in an hierarchical structure. However, methods to handle network dynamics is not considered. Furthermore, since the algorithm is centralized, it is assumed that every parameter, including the topology, link bandwidths, and link costs, are known at a central location, which is not very practical. A distributed implementation is needed for to ensure its practical applicability.

V. QoS-Based Computation with Communication Support

The idea of reducing the communicated data volume through methods like data aggregation has been recognized as a means to reduce energy consumption and prolong WSN lifetime. While majority of in-network processing proposals involve simple operations, more complex algorithms have recently been proposed to process higher data volumes such as in video sensor networks. The main idea behind such proposals is to leverage the collective processing power of individual sensor nodes to run complex processing applications. To fully utilize the collective processing power of sensor nodes, solutions from parallel processing literature have been adopted. The proposed methods also have strong connections with the communication protocols: The exchange of intermediate results occur over shared wireless channels, which is not considered in wired interconnected networks of processors. The resulting communication schedules determine channel access in sensor clusters, directly affecting the MAC layer in WSNs.

A. Collaborative Resource Allocation Algorithm [20]

The Collaborative Resource Allocation (CoRAl) algorithm [20] aims to dynamically allocate resources such as bandwidth and CPU time for multiple periodic applications in WSNs. Subject to resource availability, CoRAl aims to adjust application sampling frequencies to meet the temporal constraints and maximize network utility. CoRAl is assumed to be executed in fully-connected single-hop WSNs, where all nodes are synchronized and use Earliest Deadline First (EDF) as the scheduling algorithm. Nodes also implement the implicit EDF algorithm as the underlying wireless network MAC protocol. End-to-end applications are considered in [20] that are composed of a chain of tasks already assigned to sensors and sequentially executed in a pipelined manner.

CoRAl achieves its goals by iteratively executing the following steps until the schedule converges: First, the task execution frequencies on each sensor are locally optimized subject to application execution frequency upper-bounds, whose initial values are set to be infinite. Then the execution frequency upper-bound of each application is reevaluated based on the updated task frequencies and bandwidth allocation.

In CoRAl, the wireless channel is modeled as a dummy node on which only communication can be executed, and the network bandwidth is allocated in the same manner as sensor CPU time allocation. The CoRAl algorithm is presented in Fig. 11. In each node, an extended version of the SLSS algorithm [32] is implemented to compute locally optimal frequencies subject to node utility constraints. Different from the original SLSS algorithm, the extended SLSS algorithm in [20] takes each task's application execution frequency upper-bound into consideration. After each iteration of local optimization, the upper-bound frequency of each application is calculated. Let the *leader task* ld_i and *bottleneck task* bn_i of an application T_i be tasks whose frequency f_i^{ld} and f_i^{bn} are highest and lowest among all tasks of T_i, respectively. The frequency upper-bound of T_i is updated as $f_i^{max} = f_i^{bn} + (f_i^{ld} - f_i^{bn})\sigma$, where σ is the factor that controls frequency convergence speed. The optimization procedure terminates when the weighted difference between leader and bottleneck frequencies converges.

CoRAl addresses online resource allocation among multiple applications. According to the simulation results, CoRAl provides performances comparable to the optimal solutions obtained by the non-linear optimization tool of Matlab at a much higher execution speed. However, in CoRAl, tasks of applications are assumed to be already assigned on sensors, and task mapping remains an open problem. Furthermore, energy consumption is not explicitly considered in [20], which is a fundamental problem in WSNs.

CoRAl:

T_i:	Application i
f_i^{max}:	Maximum upper-bound frequency of application T_i
L:	Number of Applications
m_k:	Sensor node k
f_i^{ld}:	Frequency of leader task of application T_i
f_i^{bn}:	Frequency of bottleneck task of application T_i

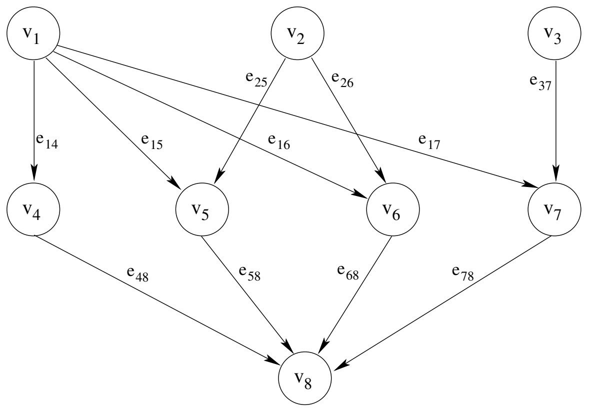
```
1. Initialize maximum upper-bound frequency of each application $T_i$:
2.     $f_i^{max} = +\infty$, $i \in \{1, ..., L\}$
3. WHILE schedule not converge
4.     FOR each sensor $m_k$
5.         FOR each task of application $T_i$ assigned on $m_k$, $i \in \{1, ..., L\}$
6.             Locally optimize the task subject to $f_i^{max}$ using the extended SLSS
7.     FOR each application $T_i$
8.         Reevaluate $f_i^{max}$ with updated $f_i^{ld}$ and $f_i^{bn}$
```

Fig. 11. The CoRAl Algorithm

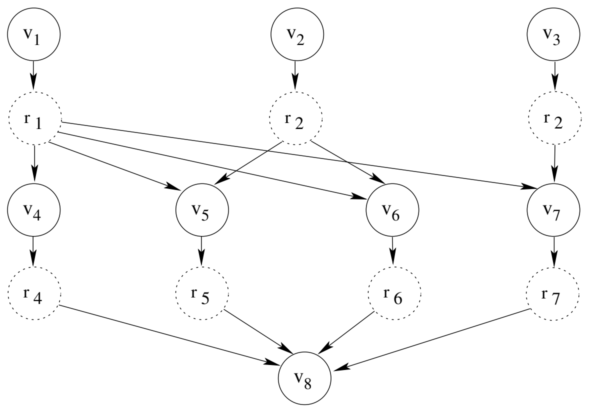
B. EcoMapS Algorithm [37]

A task mapping and scheduling solution, EcoMapS is presented in [37] for energy-constrained applications in single-hop WSNs. It is assumed that networks are composed by homogeneous sensors that can calculate and communicate simultaneously. EcoMapS aims to assign computation tasks and schedule communication events with minimum application execution lengths subject to energy consumption constraints. EcoMapS is composed of two phases: the *Initialization Phase* and the *Quick Recovery Phase*. The Initialization Phase algorithm aims to minimize schedule lengths subject to energy consumption constraints, while the Quick Recovery Phase algorithm handles run-time sensor failures.

In the Initialization Phase, EcoMapS iteratively searches for the schedule with an optimal number of *computing sensors* involved in computation that results in the minimum schedule length under the energy consumption constraint. To exploit the broadcast nature of wireless communication, a hyper-graph representation of the Directed Acyclic Graph (Hyper-DAG) is introduced. The Hyper-DAG representation of task dependency explicitly represents communication as well as computation events: The edges between a task and its immediate successors in a DAG is replaced with a *net*, which represents the communication task to send the result of a task to all of its immediate successors in the DAG. The Hyper-DAG extension of the DAG in Fig. 12(a) is presented in Fig. 12(b), where r_is are the introduced nets. Similar to CoRAl [20], EcoMapS also models the single-hop wireless channel as a virtual node where only communication tasks can be executed. Based on the virtual node model and Hyper-DAG, a communication scheduling algorithm is developed and embedded into the schedule search algorithm, E-CNPT. E-CNPT is a low-complexity algorithm that first enqueues tasks according to the critical path of a Hyper-DAG, then assigns the enqueued tasks to the node with minimum execution start time. In case communication between sensors is necessary, the proposed communication scheduling algorithm is executed. The Quick Recovery Phase algorithm handles sensor failures by adaptively adjusting the previous schedule. If idle sensors exist, the tasks of the failing sensor are migrated to an idle sensor. Otherwise, they are merged to a sensor that has the most idle time.



(a) DAG Example



(b) Hyper-DAG Extension

Fig. 12. DAG and Hyper-DAG Examples

EcoMapS is a task mapping and scheduling solution for WSNs that provides application energy consumption guarantee with minimum schedule lengths. According to the simulation results, EcoMapS has superior performance over existing mechanisms in terms of minimizing schedule lengths. Also, the alternative schedules generated after sensor failures are shown to have satisfying performance with small recovery latency. However, EcoMapS has no guarantee of application deadline constraints.

C. Energy-Balanced Task Allocation Algorithm [41]

An Energy-balanced Task Allocation (EbTA) solution is presented in [41]. EbTA assumes single-hop clustered homogeneous WSNs with multiple wireless channels, where sensors are equipped with Dynamic Voltage Scaling (DVS) enabled processors. EbTA considers real-time applications composed by inter-dependent tasks. The design objective of EbTA is to map and schedule application tasks to sensors such that balanced energy consumption is minimized subject to deadline constraints. In [41], applications are represented with Directed Acyclic Graphs (DAGs) and the scheduling problem is formulated as an Integer Linear Programming (ILP) problem. The exclusive wireless channel access feature is incorporated as additional constraints in the ILP problem.

As the formulated ILP problem is computationally costly, a three-phase heuristic is proposed in [41] to provide a practical solution. In Phase 1, tasks are grouped into clusters to minimize overall application execution time assuming infinite number of sensors. Each task first constitutes a cluster by itself. Then all communication tasks are examined in a non-increasing order of their data volume. For each communication event $e(i,j)$ between computation task T_i and T_j, the clusters containing T_i and T_j are merged if it leads to shorter application execution time. When evaluating application execution time, communication events are scheduled to the channel with smallest available time using the First Come First Serve (FCFS) policy. In Phase 2, the task clusters from Phase 1 are assigned to sensor nodes with the objective of minimizing the maximum energy expenditure among all sensors. The task clusters from Phase 1 are first sorted in a non-decreasing order of energy consumption, and stored in a queue Π. The clusters in Π are then assigned to the sensor with the minimum normalized energy consumption (task execution energy consumption normalized

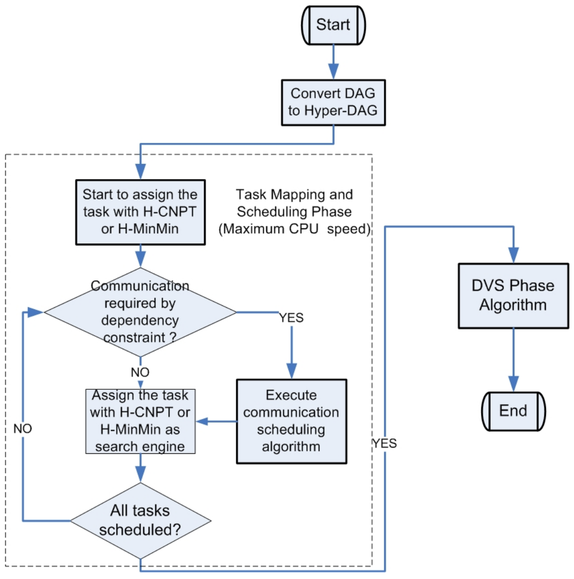

Start
Convert DAG to Hyper-DAG
Start to assign the task with H-CNPT or H-MinMin
Task Mapping and Scheduling Phase (Maximum CPU speed)
Communication required by dependency constraint ?
YES
NO
Assign the task with H-CNPT or H-MinMin as search engine
Execute communication scheduling algorithm
NO
All tasks scheduled?
YES
DVS Phase Algorithm
End

Fig. 13. Flowchart of RT-MapS

by sensor residue energy, *norm-energy* for short). Each time after a task cluster is assigned to a sensor, the norm-energy of the sensor is updated. This procedure repeats until all task clusters are assigned. Finally, a DVS heuristic is presented for Phase 3 to decrease energy consumption by iteratively adjusting the CPU voltage level of each task. In each iteration, a *critical node* that has the highest norm-energy ε is selected. Among the tasks assigned on the critical node, a task is selected such that, by decreasing its CPU supply voltage to the next level, ε is decreased the most. Each time when a task is adjusted, the application schedule is iteratively adjusted accordingly to meet inter-task dependency constraints.

EbTA is one of the first proposals that addresses task allocation in WSNs, where both communication and computation tasks are considered. It is shown through simulations that the three-phase heuristic achieves longer lifetime compared with the baseline without DVS. The performance of the three-phase heuristic is also found to be comparable to that of the ILP-based approach via simulations.

D. RT-MapS Algorithm [36]

The RT-MapS algorithm [36] is proposed for single-hop clustered WSNs, which are composed of homogeneous DVS sensors with finite number of voltage levels. The design objective of RT-MapS is to provide application deadline guarantees with the minimum energy consumption for WSNs applications. The RT-MapS algorithm contains two phases, namely, *Task Mapping and Scheduling (TMS) Phase* and *DVS Phase*. The flowchart of RT-MapS is shown in Figure 13. In the TMS phase, computation and communication events are simultaneously assigned and scheduled with the objective of minimizing energy consumption subject to deadline constraints. To guarantee deadline constraints, sensors are scheduled with highest CPU speed in the TMS phase. Schedules generated in the TMS phase are then further optimized in the DVS phase by reducing CPU speed to decrease energy consumption. Similar to EcoMapS [37], RT-MapS employs Hyper-DAGs to represent applications and utilizes the virtual node model of wireless channels.

The RT-MapS solution is outlined with the pseudo code in Fig. 14. Here, the communication scheduling

```
RT-MapS:
----------------------------------------------------------------
     $N$: Number of sensor nodes in the cluster
   $v_i$: Task $i$ of the application
   $m_k$: Sensor node $k$
   $m_k$: Sensor node $k$
$f_i^{ld}$: Frequency of leader task of application $T_i$
$f_i^{bn}$: Frequency of bottleneck task of application $T_i$
----------------------------------------------------------------

1.  Convert DAG to Hyper-DAG
2.  FOR $n$ = 0 to $N$
3.      /* Schedule with $n$ computing sensors/*
5.      Assign tasks using H-MinMin or H-CNPT
6.      IF communication is needed for an assignment of task $v_i$ on sensor $m_k$
7.          Execute the communication scheduling algorithm
8.  Among these candidate schedules, find the optimal schedule $H^o$:
9.      $H^o$ has the smallest energy consumption subject to deadline constraints
10. Adjust the schedule $H^o$ using the DVS algorithm
```

Fig. 14. The RT-MapS Algorithm

algorithm sequentially schedules communication tasks on the virtual channel node to avoid packet collision. Broadcasting is also realized in the communication scheduling algorithm to conserve energy. The communication scheduling algorithm is embedded in the execution of the task mapping and scheduling algorithms, H-CNPT and H-MinMin. H-CNPT is different from E-CNPT [37]. Among schedules with different number of computing sensors, the schedule satisfying the deadline constraint with the minimum energy consumption is selected as the optimal solution. The H-MinMin algorithm is an extended version of the Min-Min algorithm [33]. In the core of H-MinMin lies the fitness function that combines schedule length and energy consumption resulting from assigning a task to a sensor. In each iteration of task assignment, each "mappable" task whose immediate predecessors are already assigned is tentatively assigned to different sensors. The sub-optimal task-sensor combination with minimum fitness value is kept. Among all sub-optimal task-sensor combinations, the pair that gives the minimum fitness value is chosen, and the task is assigned to the corresponding sensor. The pseudo code of H-MinMin is presented in Figure 15. The DVS algorithm further reduces energy consumption of the schedules generated in the TMS phase. In the DVS algorithm, the communication tasks assigned on the channel are kept unchanged, and their start time and finish time are taken as the upper and lower bounds to adjust the corresponding sensors' speed during the time interval. During DVS adjustment procedure, CPU speed is reduced in proportion to the CPU utility.

RT-MapS aims to provide application deadline guarantees with minimum energy consumption. Due to the parallelism among sensors and exploiting the broadcast feature of wireless communication, RT-MapS shows superior performance compared with exiting mechanisms including EbTA [41], as presented in the simulation part of [36]. However, as other outlined solutions, RT-MapS also fails to extend to multi-hop WSN cluster. Furthermore, execution of concurrent application, and interaction with neighboring clusters are not considered, either.

```
H-MinMin:
    $L$: Mappable task list
    $v_i$: Task $i$ of the application
    $m_k$: Sensor node $k$

1.  FOR $\alpha$ = 0 step 0.1 to 1
2.      Assign all entry-tasks
3.      Initialize the mappable-task list $L$
4.      WHILE $L$ is not empty
5.          FOR task $v_i \in L$
6.              Find sensor $m_k$ with the minimum fitness of $v_i$
7.          Find $(v^o, m^o)$ with the minimum fitness among these combinations
8.          Assign $v^o$ to $m^o$
9.  Update $L$
10. Among all schedules with different values of $\alpha$
11.     Select the optimal schedule
```

Fig. 15. The H-MinMin Algorithm used in the RT-MapS Algorithm

VI. QoS-Based Capacity Estimation in WSNs

An important step in end-to-end QoS provisioning in any communication network is the estimation of the network capacity to ensure that admitted flows can be sustained in the network. Such estimations are also important to adjust the data injection rates and other requirements of flows at sources. To provide accurate feedback, sinks must be able to compute the capacity of the network as functions of QoS parameters and the flow characteristics. In the literature, capacity modeling of wireless multihop networks have been investigated with simple metrics. The line of work pioneered by P.R. Kumar aims wireless network capacity in bit-meters per second [21]. This approach has also stimulated similar analysis attempts with increasingly realistic communication models [39], [4], [35], [22]. The main shortcoming of these attempts lies in the oversight of an overarching goal of communication networks: Delivery of information to remote devices across a network under realistic conditions. Furthermore, combined metrics do not represent individual metrics such as delay, throughput, and loss rates explicitly.

Motivated by the lack of multi-metric capacity modeling work for multihop networks, we have investigated the following problem [3]: *Consider a dense sensor network deployed in a rectangular area, where sources and destinations are located along two opposite edges. Assuming a Rayleigh fading channel, what is the maximum attainable throughput across the network given a delay bound for individual packets and a maximum tolerable bit error rate?* A corollary to this problem is the calculation of the minimum delay attainable if a particular throughput is required. With the results of this analysis, it is possible to estimate the total capacity of the network for a given delay value.

Our analysis is based on a physical layer channel model that accounts for the effects of the noise as well as the interference from other simultaneous transmissions. The bit error rates are calculated as the probability of a signal to be below a given threshold based on the Rayleigh channel model. With this comprehensive interference model, we analyze the effect of packets in the same data flow (intra-flow interference) and other data flows (inter-flow interference).

As a first step, we analyzed a data stream that traverses a linear path. Based on a discrete time model where each packet transmission lasts one time unit, we modeled the behavior of the packets in the same flow. Our analysis confirmed that every packet injected into the network slows down the progress of preceding packets. To solve this problem, waiting times are introduced at the source while injecting packets into the network, allowing earlier packets to move forward before another packet (i.e., an interferer) is introduced. To achieve the lowest end-to-end delay, a packet must exist in the network alone, which reduces the throughput to one packet per end-to-end delay. Similarly, the maximum throughput can be achieved if packets are injected every time unit, which causes the delay to go to infinity as the network density approaches infinity. Intermediate combinations are achievable by adjusting the inter-packet waiting times.

The two-dimensional case is modeled as parallel linear flows. In this case, every transmitted packet experiences inter-flow as well as intra-flow interference. To increase the throughput, the distance between flows can be reduced. Small inter-flow separations increases the inter-flow interference, causing packets to cover short distances every transmission time, which in turn increases the end-to-end delay. If the flows are separated by larger distances, then the overall throughput decreases along with the end-to-end delay. The parameters that control this behavior are the inter-packet delays, relative packet injection times between flows, and the inter-flow separations. Using non-linear optimization techniques, it is possible to compute the maximum throughput for a given delay bound and parameters to achieve the desired performance.

This study [3] is merely a first step in this open research area, and leaves out many important issues such as crossing paths, single destination flows, and non-time-synchronized systems. Furthermore, protocol dependent effects on capacity are also left out for the sake of simplicity. We are hopeful that more accurate and comprehensive studies will follow this one that will address more realistic settings and that can be directly integrated to admission control and QoS feedback mechanisms.

VII. Conclusions and Open Research Problems

QoS-based communication in WSNs is a very promising and emerging field to sustain the communication in next wave of truly real-time WSNs. Although there is a significant amount of work that has been performed in the last five years in this area, they are far from addressing all QoS problems in WSNs. While WSNs resemble ad hoc networks and many more solutions for ad hoc networks do exist, it is important to keep in mind that WSNs are fundamentally different than ad hoc networks and have different requirements and constraint. Yet, it is equally important to consider the lessons learned from earlier QoS-based communication protocol proposals to avoid similar pitfalls.

Considering the available pool of solutions, several shortcomings in the development of QoS-based communication protocols stand out: Most of the solutions either rely on hard-to-materialize assumptions or are very complex for implementation on truly resource-constrained sensor nodes. Although the existing and widely used sensor nodes (such as Mica2 sensor nodes [13]) would not have any problems running algorithms outlined in this chapter, their implementation on extremely small devices is questionable. Therefore, simplification of the QoS support mechanisms is very important. Similarly, implementation of select proposals on very simple devices should also be undertaken to assess their value in real operation environments.

A majority of solutions consider only an isolated set of problems (such as only routing, MAC, or processing) or very few QoS parameters (such as delay, reliability, or energy). While the protocol design for WSNs is heavily influenced by applications, protocols that can support multiple metrics are still few and far apart. Support of multiple QoS metrics is an important step to realize many real-time and mission-critical applications. It is clear that such protocols would have to coordinate (if not merge) several networking functions traditionally considered belonging to separate layers. The resulting solutions must still maintain a simple nature for easy and ubiquitous deployment on many platforms.

Another important aspect that has not been studied very extensively is the feedback mechanisms. The solutions presented in this chapter operate in an open loop; no feedback is obtained from end nodes to adjust the behavior of the flows or protocols. Although some examples such as ESRT [31] use feedback mechanisms to sustain specific reporting requirements, they are mostly application-specific solutions and do not apply to generic settings. Furthermore, feedback mechanisms to adjust protocol behavior would improve both the application level QoS guarantees as well as improve the total number of applications supported in a real-time WSN.

Finally, the QoS-based capacity estimation is a completely virgin field with very limited work present. Capacity estimation of multihop wireless networks is a challenging topic in itself and very few proposals can provide actual limits (and not only scaling laws) even with very simplifying assumptions. Clearly, capacity estimation has no low-hanging fruits. Significant and coordinated research efforts are required to derive useful capacity bounds. These bounds should not only provide idealized limits, but also useful estimates that can be used in combination with QoS-based protocols.

VIII. Exercises

1) One of the most important reasons for performance degradation in wireless networks in general is the mismatch between solutions deployed together. Considering the protocols and solutions outlined in this chapter, identify
   a) Protocols of different layers that cannot be used in the same node, and
   b) Protocols not necessarily in different layers that cannot be run concurrently in the same network.

   Elaborate on reasons for these incompatibilities. Expand your search to more recent work published in the literature.
2) Considering the SPEED protocol, how would the selection of the SetSpeed parameter affect the system performance? Based on these observations, propose a method to select the SetSpeed value for a given overall throughput requirement.
3) Repeat the exercise above for the MMSPEED protocol.
4) How should the RT-MapS algorithm be modified such that it can be used in multi-hop clusters? Can the presented channel model be sufficient? If not, how should it be updated?
5) The ESRT protocol aims to keep congestion under control by changing the event reporting rates. Identify routing (and MAC if necessary) protocols that are best suited to be used with ESRT. Discuss possible needs for modification for seamless operability. Elaborate on the multihop feedback channel on ESRT performance, and how routing (and MAC) protocols can help with potential problems.
6) Discuss how energy-awareness can be incorporated into multi-path routing protocols in WSNs.

References

[1] K. Akkaya and M. Younis. An Energy-Aware QoS Routing Protocol for Wireless Sensor Networks. *Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems*, 2003.

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaninam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks Journal (Elsevier)*, 38(4):393–422, March 2002.

[3] A. Bader and E. Ekici. Throughput and Delay Optimization in Interference-Limited Multi-Hop Networks. *Proceedings of Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2006)*, May 2006.

[4] G. Barrenechea, B. Beferull-Lozano, and M. Vetterli. Lattice Sensor Networks: Capacity Limits, Optimal Routing and Robustness to Failures. *Proceedings of Third International Symposium on Information Processing in Sensor Networks, IPSN 2004*, April 2004.

[5] R. Benkoczi, H. Hassanein, S. Akl, and S. Tai. Qos for data relaying in hierarchical wireless sensor networks. *Proceedings of the 1st ACM International Workshop on Quality of Service and Security in Wireless and Mobile Networks*, pages 47–54, 2005.

[6] D. Bertsekas and R. Gallager. *Data Networks*. Prentic Hall, 1987.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. *RFC 2475*, December 1998.

[8] A. Boukerche, X. Cheng, and J. Li. A performance evaluation of a novel energy-aware data-centering algorithm for wireless sensor networks. *ACM/Springer Wireless Networks*, pages 619–636, 2005.

[9] A. Boukerche and H. Owens. Energy-aware routing protocol for mobile and wireless ad hoc networks. *Proceedings of IEEE Conference on Local Networks*, pages 768–771, 2003.

[10] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: An Overview. *RFC 1633*, June 1994.

[11] J. Campbell, P. Gibbons, S. Nath, S. Seshan, and R. Sukthankar. IrisNet: An Internet-scale Architecture for Multimedia Sensors. *Proceedins of ACM Multimedia Conference*, 2005.

[12] S. Chen and K. Nahrstedt. An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions. *IEEE Network*, 12(6):64–79, November/December 1998.

[13] Crossbow, Inc. MICA2DOT Data Sheet. *http://www.xbow.com*, 2006.

[14] M. de Prycker. *Asynchronous Transfer Mode: Solutions for Broadband ISDN*. Prentice Hall, 3^{rd} edition, 1995.

[15] B. Deb, S. Bhatnagar, and B. Nath. ReInForm: Reliable Information Forwarding Using Multiple Paths in Sensor Networks. *Proceedings of IEEE International Conference on Local Computer Networks*, pages 406–415, 2003.

[16] E. Ekici. *Resource Management in Wireless Networking*, chapter QoS-Based Routing in Wireless Mobile Networks. Kluwer, 2005.

[17] A. Felemban, C.G. Lee, and E. Ekici. MMSPEED: Multi-Path Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 5(6):738–754, June 2006.

[18] W. Feng, B. Code, E. Kaiser, W. Feng, and M. L. Baillif. Panoptes: Scalable Low-Power Video Sensor Networking Technologies. *ACM Transactions on Multimedia Computing, Communications, and Applications*, January 2005.

[19] M. Gerla and K. Xu. Multimedia Streaming in Large-Scale Sensor Networks with Mobile Swarms. *ACM SIGMOD Record*, December 2003.

[20] S. Giannecchini, M. Caccamo, and Chi-Sheng Shih. Collaborative resource allocation in wireless sensor networks. In *Proc. of Euromicro Conference on Real-Time Systems (ECRTS'04)*, pages 35–44, June/July 2004.

[21] P. Gupta and P.R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[22] M. Haenggi. Toward a circuit theory for sensor networks with fading channels. *Proceedings of the 2004 International Symposium on Circuits and Systems ISCAS '04*, 4(IV):908–911, May 2004.

[23] T. He, J.A. Stankovic, C. Lu, and T. Abdelzaher. SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks. *Proceedings of IEEE International Conference on Distributed Computing Systems*, pages 46–55, 2003.

[24] D.B. Johnson, D.A. Maltz, Y. Hu, and J.G. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks (dsr). *Internet Draft, draft-ietf-manet-dsr-07.txt*, 2002.

[25] H.S. Kim and T.F. Abdelzaher. Dynamic delay-constrained minimum-energy dissemination in wireless sensor networks. *ACM Transactions on Embedded Computing Systems*, 4(3):679–706, August 2005.

[26] Huan Li, Prashant Shenoy, and Krithi Ramamritham. Scheduling communication in real-time sensor applications. In *Proc. of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'04)*, pages 10–18, May 2004.

[27] Yang Liu, Itamar Elhanany, and Hairong Qi. An energy-efficient qos-aware media access control protocol for wireless sensor networks. In *Proc. of International Conference on Mobile Adhoc Sensor Systems (MASS'05)*, pages 189–191, November 2005.

[28] Jelena Misic, Shairmina Shafi, and Vojislav B. Misic. Maintaining reliability through activity management in 805.15.4 sensor networks. In *Proc. of 2nd International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine'05)*, August 2005.

[29] A. parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. In *Proc. of Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'92)*, 1992.

[30] V.D. Park and M.S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. *Proceedings of IEEE INFOCOM*, pages 1405–1413, 1997.

[31] Y. Sankarasubramaniam, O.B. Akan, and I.F. Akyildiz. ESRT: Event-to-Sink Reliabile Transport in Wireless Sensor Networks. *Proceedings of the Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM MobiHoc 2003)*, 2003.

[32] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin. On task schedulability in real-time control systems. In *Proc. of the 17th IEEE Real-Time Systems Symposium (RTSS'96)*, December 1996.

[33] S. Shivle, R. Castain, H. J. Siegel, A. A. Maciejewski, T. Banka, K. Chindam, S. Dussinger, P. Pichumani, P. Satyasekaan, W. Saylor, D.Sendek, J. Sousa, J. Sridharan, P. Sugavanam, and J. Velazco. Static mapping of subtasks in a heterogeneous ad hoc grid environment. In *Proceedings of IEEE Parallel and Distributed Processing Symposium*, April 2004.

[34] K. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, pages 16–27, October 2000.

[35] E.S. Sousa and J.A. Silvester. Optimum transmission ranges in a direct-sequence spread-spectrum multihop packet radio network. *IEEE Journal on Selected Areas in Communications*, 8(5):762–771, June 1990.

[36] Y. Tian, J. Boangoat, F. Özgüner, and E. Ekici. Real-Time Task Mapping and Scheduling for Collaborative In-Network Processing in DVS-Enabled Wireless Sensor Networks. *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, April 2006.

[37] Y. Tian, E. Ekici, and F. Özgüner. Energy Constrained Local Task Mapping and Scheduling in Wireless Sensor Networks. *Proceedings of the First IEEE International Workshop on Resource Provisioning and Management in Sensor Networks (RPMSN05)*, November 2005.

[38] P. White. RSVP and Integrated Services in the Internet: A Tutorial. *IEEE Communications Magazine*, 35:100–106, May 1997.

[39] F. Xue, L.-L. Xie, and P.R. Kumar. The transport capacity of wireless networks over fading channels. *IEEE Transactions on Information Theory*, 51(3), March 2005.

[40] M. Younis, K. Akkaya, M. Eltoweissy, and A. Wadaa. On Handling QoS Traffic in Wireless Sensor Networks. *Proceedings of the* 37^{th} *IEEE Annual Hawaii International Conference on System Sciences (HICSS'04)*, pages 292–301, January 2004.

[41] Y. Yu and V. K. Prasanna. Energy-balanced task allocation for collaborative processing in wireless sensor networks. *ACM/Kluwer Journal of Mobile Networks and Applications*, 10(1-2):115–131, February 2005.